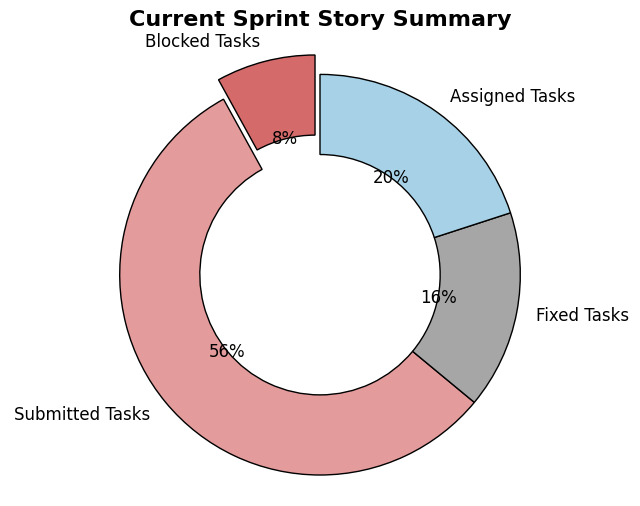
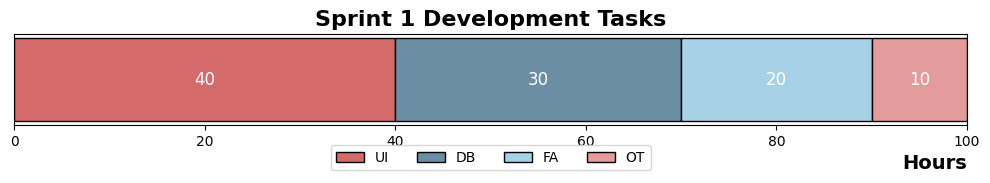
**Sprint Report**

**Sprint 1 (Weeks 2-5) Report**

1. **Sprint Overview**
   1. **Sprint ID**: 2024-9
   2. **Start Date**: 2024.9.9
   3. **End Date**: 2024.9.30
   4. **Team Size**: 8 members
   5. **Estimated Workload**: 100 hours
2. **Sprint Goals**: Establish basic functionality and architecture to meet fundamental requirements.
3. **Completed Work**
   1. **User Stories/Tasks**:
      1. **User Interface Module**
         1. **Description**: Implemented the file upload submodule, allowing users to upload Python project files through the front-end interface. Developed the initial interface that displays an overview of CWE issues and code quality scores.
         2. **Story Points**: 15
      2. **Database Module**
         1. **Description**: Completed the preliminary design of the database architecture and created a module to store records of user project submissions.
         2. **Story Points**: 10
   2. **Acceptance Criteria**:
      1. **User Interface Module**:
         1. The file upload function works correctly across different browsers.
         2. The overview page accurately displays the number of CWE issues and the code quality score for the uploaded projects, and the design meets user-friendliness standards.
      2. **Database Module**:
         1. The database design aligns with system architecture standards, storing user project submission records, including the number of CWE issues and code quality scores.
         2. Stored data has been validated through basic queries to ensure accuracy and stability.
4. **Incomplete Work**
   1. **Unfinished Items**:
      1. **File Analysis Module**
         1. **Challenges**: The file analysis module was not fully completed, particularly the complex error detection and handling sections. This part involved multi-layer nested structures, recursive calls, and dynamically generated code, which made detecting errors challenging for Pylint. As a result, certain complex CWE issues (such as undefined variables in dependency chains or dynamic type conversion issues) were not handled successfully, and the generated analysis report was incomplete.
         2. **Performance Bottlenecks**: When analyzing large Python codebases, Pylint's performance lagged, causing delays, especially with projects containing numerous external dependencies and complex logic.
   2. **Deferred Items**:
      1. **File Analysis Module**: The unresolved error detection aspects of the file analysis module have been postponed to the next sprint. The development team plans to focus on the following issues in the next iteration:
         1. Optimize Pylint's performance to better support large-scale code analysis.
         2. Introduce additional static analysis tools or perform manual analysis to address complex errors related to cross-module dependencies and multi-layer nested code, ensuring the completeness and accuracy of the CWE analysis report.
5. **Challenges & Solutions**
   1. **Visualization Interface Incomplete, Insufficient Data Presentation Clarity**:
      1. **Challenge**: While developing the user interface module, the team faced design challenges in visualizing CWE issues and code quality scores in an intuitive and user-friendly way.
      2. **Solution**: After multiple discussions with the front-end designer and referencing mature visualization design examples, the team decided to improve data presentation by incorporating clearer chart types, such as bar and pie charts, to represent the overview of CWE issues and code quality scores.
   2. **Inefficient Database Table Structure with Redundant Data**:
      1. **Challenge**: During the database module design, the team identified redundancy in the initial database table structure, which affected query performance, especially with large datasets, causing slow system response times.
      2. **Solution**: The team collaborated with the system architect to review the database design, simplifying the table structure and removing redundant fields. By adopting a normalized design, they minimized data duplication and optimized query performance. Additionally, the team decided to conduct database performance testing in the next sprint to ensure that the further optimized design meets system requirements.
6. **Review Feedback**
   1. **Summary of Sprint Review Feedback**:
      1. **Communication and Collaboration Deficiencies**: During Sprint 1, there was insufficient communication among team members regarding the details of system architecture and module implementation, leading to inconsistencies during integration. Specifically, the integration between the front end and database suffered from a lack of coordination, causing redundant work and reduced efficiency.
      2. **Insufficient Test Coverage**: The team noted that although basic functionalities were established, there was a lack of comprehensive unit testing for each module, increasing the risk of issues arising during subsequent integration.
   2. **Areas for Improvement**:
      1. **Enhancing Communication Mechanisms**: It is recommended to increase team meetings, such as weekly module progress discussions, in the next sprint to ensure all members have a consistent understanding of project updates, reducing conflicts and rework during integration.
      2. **Increasing Test Coverage**: Moving forward, the team will focus more on designing and implementing unit and integration tests to ensure each function is thoroughly tested post-implementation, reducing the number of bugs and enhancing overall system stability.
7. **Action Items for Next Sprint**
   1. **Optimization and Enhancement of File Analysis Module**
      1. **Pylint Performance Optimization**: Address the performance bottleneck with Pylint when processing large codebases to improve response times for projects with numerous external dependencies and complex logic.
      2. **Error Detection Improvement**: Continue refining the file analysis module for complex errors in multi-layer nested structures, recursive calls, and dynamically generated code. Introduce additional static analysis tools or manual analysis to supplement Pylint, ensuring comprehensive and accurate CWE issue analysis.
      3. **Development of Scoring Calculation Module**: Create a scoring calculation module based on code analysis results to provide a more comprehensive code quality score.
   2. **Further Optimization of Database Module**
      1. **Database Performance Testing**: Conduct performance testing on the modified database structure to ensure that the removal of redundant data meets expected system query performance.
      2. **Implementation of User Record Query Module**: Develop a module allowing users to view their project history and analysis results upon login, ensuring secure and effective data access.
   3. **Development of ChatGPT Interaction Module**
      1. **CWE Issue Transmission Module**: Develop and refine the module that sends detected CWE issues to ChatGPT, ensuring ChatGPT can provide accurate Chinese modification suggestions and return them to the system interface for an improved user experience.
      2. **Response Processing Optimization**: Optimize the ChatGPT response processing module to allow users to easily view modification suggestions and the corresponding code versions, enhancing compatibility across modules.
   4. **Code Quality Improvement and Test Coverage Enhancement**
      1. **Code Quality Improvement**: Based on Sprint 1 feedback, improve documentation and code comments, ensuring all modules conform to a unified code style standard (PEP 8) to enhance code readability.
      2. **Test Coverage Enhancement**:
         1. **Unit and Integration Testing**: Design and implement unit and integration tests for each module (e.g., file analysis, user interface, database) to ensure functionality stability.
         2. **Test Coverage Tools**: Use tools to check test coverage, prioritizing critical functionality modules to ensure coverage meets project requirements.
   5. **Strengthening Communication and Collaboration Mechanisms**
      1. **Regular Team Meetings**: Add a weekly module progress meeting to ensure all members have a consistent understanding of project changes and progress, reducing integration issues caused by insufficient communication.
      2. **Cross-Review Mechanism**: Implement cross-review after each development task to ensure each developer gains a basic understanding of other modules, enhancing overall team collaboration.
   6. **User Experience Improvement**
      1. **Data Visualization Interface Enhancement**: Based on feedback from Sprint 1, further optimize the UI by adding more intuitive chart types (e.g., bar charts, pie charts) to enhance the user experience when displaying CWE issues and code quality scores.
8. **Current Week Story Summary**



1. **Sprint1 Development Tasks**



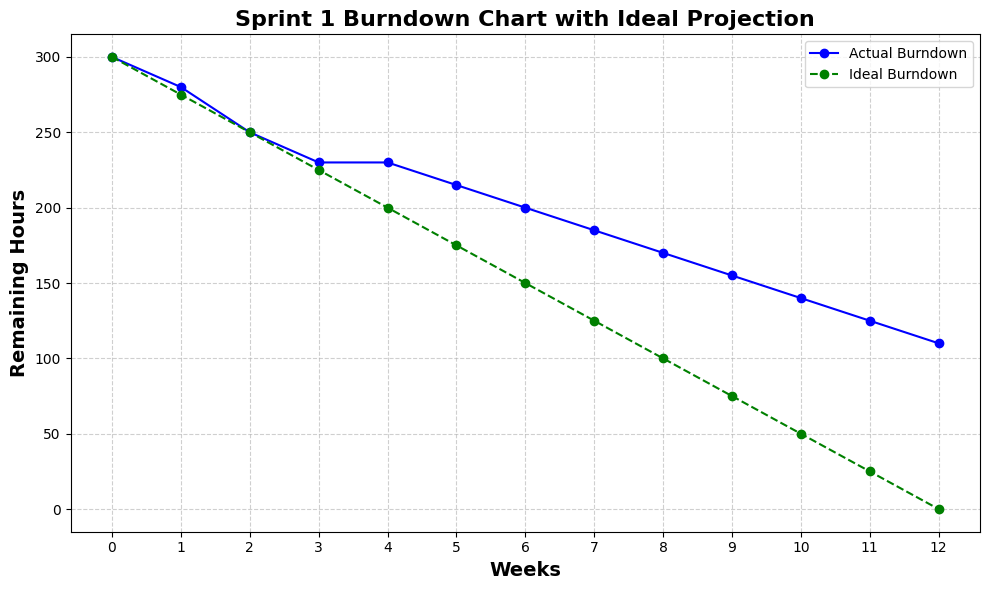
User Interface Development Task (UI)

Database Task (DB)

File Analysis Task (FA)

Other Task (OT)

1. **Sprint 1 Burndown Chart with ideal Projection**



**Sprint 2 (Weeks 6-9) Report**

1. **Sprint Overview**
   1. **Sprint ID**: 2024-10
   2. **Start Date**: 2024.9.30
   3. **End Date**: 2024.10.30
   4. **Team Size**: 8 members
   5. **Estimated Workload**: 100 hours
2. **Sprint Goals**: Improve code quality and complete the primary analysis and feedback functions.
3. **Completed Work**
   1. **User Stories/Tasks**:
      1. **File Analysis Module Optimization**
         1. **Description**: Completed initial performance optimization for Pylint, significantly improving response times when handling code with extensive external dependencies and complex logic.
         2. **Story Points**: 20
      2. **Database User Record Query Module**
         1. **Description**: Implemented the database user record query module, allowing users to log in and view historical data and analysis results for their projects.
         2. **Story Points**: 15
      3. **ChatGPT Interaction Module Development**
         1. **Description**: Completed the CWE issue transmission module, which sends detected CWE issues to ChatGPT and successfully receives modification suggestions in Chinese.
         2. **Story Points**: 10
      4. **Code Quality Improvement**
         1. **Description**: Enhanced documentation and code comments for all modules, ensuring code meets formal coding standards and improving readability.
         2. **Story Points**: 10
   2. **Acceptance Criteria**:
      1. **File Analysis Module Optimization**: Pylint’s response time reduced by 30% when handling code with complex logic and numerous external dependencies, with partial handling of errors in nested structures and recursive calls.
      2. **Database User Record Query Module**: Users can log in and successfully query project history, with query performance tested to meet expected response times.
      3. **ChatGPT Interaction Module**: CWE issues can be successfully sent to ChatGPT, and the system correctly displays ChatGPT’s returned modification suggestions in Chinese.
      4. **Code Quality Improvement**: All code meets coding standards and passes static analysis tool checks with no critical warnings.
4. **Incomplete Work**
   1. **Unfinished Items**:
      1. **Scoring Calculation Module Development**: The scoring calculation module is incomplete; only part of the scoring logic has been implemented, and final overall scores cannot be generated. The delay is mainly due to a lack of unified scoring standards across different modules, which slowed down integration.
      2. **ChatGPT Response Processing Module Optimization:**

response processing optimization in the ChatGPT interaction module was not fully completed. Users receive suggestions, but improvements in the interface display and user interaction are still needed. The delay is due to insufficient testing on user experience optimization, requiring further refinement.

* 1. **Deferred Items**:
     1. **Scoring Calculation Module**: Development has been postponed to the next Sprint to ensure consistency and completeness in scoring logic across modules.
     2. **ChatGPT Response Processing Optimization**: Further optimization of the ChatGPT response processing module will continue in the next Sprint to enhance user experience, ensuring users can easily view and apply ChatGPT’s suggestions.

1. **Challenges and Solutions**
   1. **Interface Display Issues for the ChatGPT Interaction Module**
      1. **Challenge**: Although the ChatGPT interaction functionality was implemented, users reported that the interface experience was not user-friendly. Specifically, users found it difficult to locate and apply relevant suggestions from the returned responses conveniently.
      2. **Solution**: The team made partial adjustments to the ChatGPT module, optimizing the visualization of returned suggestions to provide a more intuitive feedback interface. Additionally, user experience testing is planned for the next sprint to gather further improvement suggestions.
   2. **Cross-Origin Issues**
      1. **Challenge**: During testing of the front-end and back-end interaction between Vue and Django, it was discovered that due to inconsistencies in "protocol, domain name, and port," browsers blocked cross-origin requests. This cross-origin issue severely disrupted the data interaction process.
      2. **Solution**: To resolve the cross-origin issue, the front-end configured proxy forwarding in the config/index.js file of the Vue project to ensure that requests could be properly forwarded to the back-end while maintaining consistency in "protocol, domain name, and port." On the back-end, the django-cors-headers module was introduced, and related cross-origin access configurations were enabled in the settings.py file, allowing cross-origin requests from the front-end domain. This collaborative front-end and back-end solution successfully bypassed browser cross-origin restrictions, ensuring normal data transmission.
   3. **Data Format Parsing Issues**
      1. **Challenge**: The back-end encountered problems in correctly parsing the content from request.body or request.POST.data when receiving requests from the front-end, resulting in empty request data. This directly hindered the back-end's ability to process and respond to data, affecting functionality.
      2. **Solution**: To ensure that data formats could be correctly parsed, the front-end standardized the use of JSON format for transmitting request content and set the Content-Type: application/json header. On the back-end, Django's JsonResponse library was used to parse JSON data, ensuring the content sent from the front-end could be correctly extracted and processed. This adjustment improved the reliability of data transmission between the front-end and back-end, providing a solid foundation for functionality.
2. **Review Feedback**
   1. **Summary of Sprint Review Feedback**:
      1. **Insufficient Automated Test Coverage**: In Sprint 3, while the team completed most module integration tests, automated test coverage remains lacking, especially in scenarios involving complex business logic and system load. The delay in developing some automated testing tools impacted the overall system stability verification.
      2. **Challenges in Module Integration**: During module integration, particularly with the scoring calculation and ChatGPT response processing modules, the team encountered inconsistencies. These modules’ dependencies and complex communication resulted in unexpected behaviors after integration, causing delays and increasing testing and debugging workloads.
      3. **Further Improvements Needed in User Experience**: Despite multiple optimizations for the ChatGPT interaction module interface, some user feedback indicated unresolved issues, particularly in handling multiple modification suggestions. Users found the interface guidance unclear when managing several suggestions, making understanding and application challenging.
   2. **Areas for Improvement**:
      1. **Increasing Team Flexibility**: The team should enhance its flexibility in managing varying workloads in future development, particularly by promptly adjusting priorities when certain modules encounter challenges, ensuring that critical modules do not delay overall progress.
      2. **Strengthening Module Integration Planning**: To address integration issues, the team will focus more on module integration timing in the next Sprint. It is recommended to conduct integration tests immediately after completing each module to identify and resolve inconsistencies early.
      3. **User Experience Enhancement**: Particularly for the ChatGPT interaction module’s UI display, it is recommended to conduct user experience testing in the next Sprint, gather feedback from actual users, and make further optimizations to help users locate and apply modification suggestions more easily.
3. **Action Items for Next Sprint**
   1. **Testing and Debugging**
      1. **Integration Testing**: Conduct comprehensive integration testing across all modules (file analysis, ChatGPT interaction, and user interface modules) to ensure seamless communication. Pay special attention to the postponed scoring calculation and ChatGPT response processing modules, ensuring they integrate smoothly.
      2. **Develop Automated Testing Tools**: Build automated testing tools focusing on validating module functionality and overall system stability, increasing test coverage.
   2. **Development of History Tracking and Visualization Module**

**CWE Issue Trend Chart and Score Change Chart**: Implement modules for CWE issue trends and score changes to help users visualize changes in project security issues and code quality over time. Ensure the history tracking module meets functional and user experience expectations.

* 1. **Development of Export and Report Generation Module**

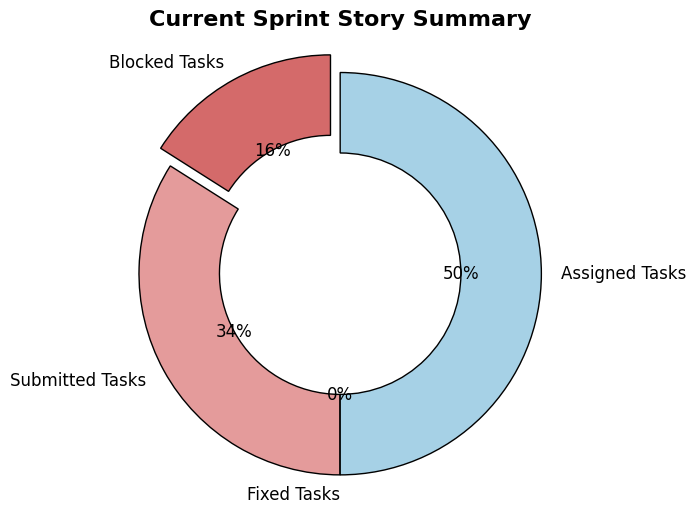
**Data Export Functionality**: Develop an export module allowing users to export project history and analysis reports in CSV or PDF format for future reference and analysis, focusing on data integrity and readability.

* 1. **Development of User Authentication and Account Management Module**

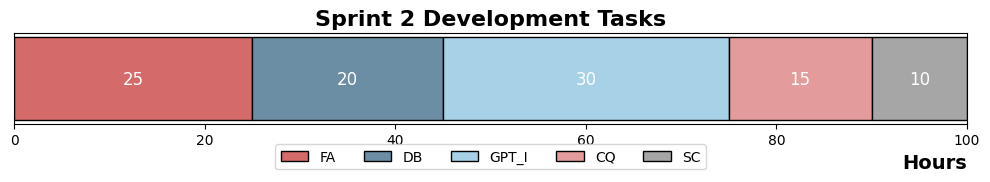
**User Registration and Login Functionality**: Implement the user authentication module, providing registration and login functionality to ensure secure access and management of users’ project records, enhancing system security.

* 1. **User Experience Improvement**
     1. **Optimization of ChatGPT Interaction Module**: Continue optimizing the ChatGPT response processing module’s interface display, conduct user experience testing to gather feedback, and further improve the interface design. Ensure users can easily locate and apply modification suggestions to their projects.
     2. **Data Visualization Enhancements**: Based on user feedback, further improve the UI interface, optimizing data display methods (e.g., bar and pie charts) to enhance user experience when using CWE issue and code quality score display functions.
  2. **Strengthening Communication and Collaboration Mechanisms**
     1. **Regular Team Meetings**: Continue holding weekly module progress meetings to ensure team members have a consistent understanding of project progress and changes, reducing communication gaps during development.
     2. **Cross-Review Mechanism**: Strengthen the cross-review process for modules, ensuring each developer has a basic understanding of other modules, further enhancing team collaboration.
  3. **Project Demonstration and Delivery Preparation**
     1. **Create Project Demonstration Video**: Produce a demonstration video showcasing system functionality and module implementation, helping stakeholders understand project progress and feature implementation.
     2. **Project Submission and Review**: Prepare and submit the GitHub project link for external team review and testing, ensuring the project meets delivery requirements and is ready for final release.

1. **Current Sprint Story Summary:**



1. **Sprint2 Development Tasks**



File Analysis Task (FA)

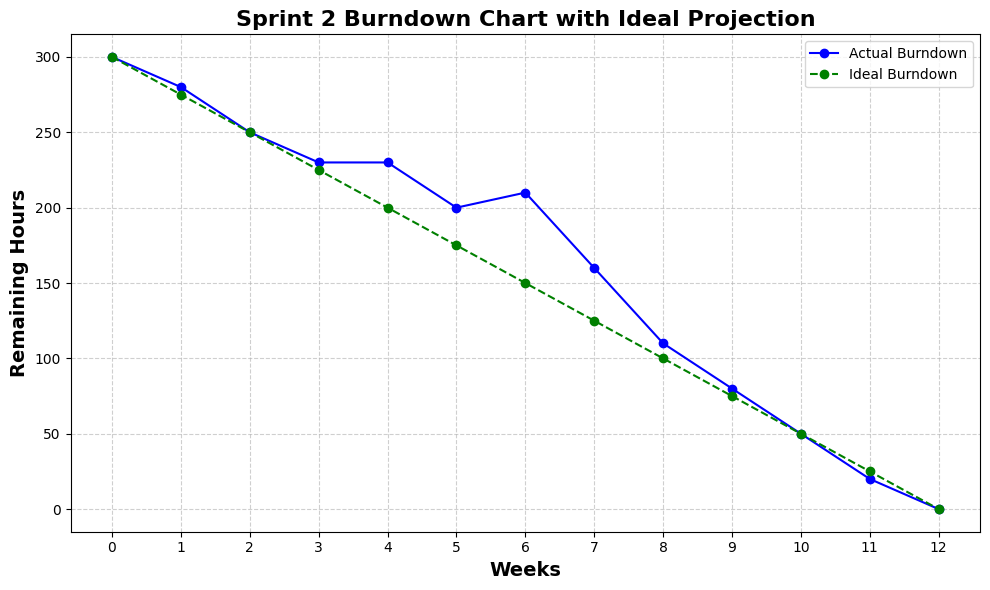
Database Task (DB)

ChatGPT Interaction Module Development Task (GPT\_I)

Code Quality Improvement Task (CQ)

Scoring Calculation Module Development (SC)

1. **Sprint2 Burndown Chart with Ideal Projection**



**Sprint 3 (Weeks 10-13) Report**

1. **Sprint Overview**
   1. **Sprint ID**: 2024-11
   2. **Start Date**: 2024.10.30
   3. **End Date**: 2024.11.30
   4. **Team Size**: 8 members
   5. **Estimated Workload**: 120 hours
2. **Sprint Goals**: Automated testing, system integration, and final delivery.
3. **Completed Work**
   1. **User Stories/Tasks**:
      1. **History Tracking and Visualization Module Development**
         1. **Description**: Implemented CWE issue trend and score change modules to help users intuitively understand changes in project security issues and code quality.
         2. **Story Points**: 20
      2. **Export and Report Generation Module Development**
         1. **Description**: Completed the data export functionality, allowing users to export project history and analysis reports as CSV or PDF files, ensuring data integrity and readability.
         2. **Story Points**: 15
      3. **User Authentication and Account Management Module Development**
         1. **Description**: Developed the user registration and login module, allowing users to securely access and manage their project records.
         2. **Story Points**: 15
      4. **User Experience Improvements for ChatGPT Interaction Module**
         1. **Description**: Optimized the ChatGPT response processing module interface and conducted user experience testing to gather actual user feedback. Ensured users could easily locate modification suggestions and apply them to projects.
         2. **Story Points**: 10
      5. **Data Visualization Improvement**
         1. **Description**: Further optimized the UI interface based on user feedback, improving data presentation methods (e.g., bar and pie charts) to enhance the user experience with CWE issue and code quality score display features.
         2. **Story Points**: 10
      6. **Project Demonstration and Delivery Preparation**
         1. **Description**: Created a project demonstration video showcasing module functionalities and implementations. Also organized and submitted the GitHub project link for external team review and testing.
         2. **Story Points**: 10
   2. **Acceptance Criteria**:
      1. **History Tracking and Visualization Module**: The CWE issue trend and score change charts correctly display historical data, allowing users to intuitively understand changes in project security and code quality.
      2. **Export and Report Generation Module**: Project history data and analysis reports can be successfully exported as CSV or PDF files, with complete and readable contents.
      3. **User Authentication and Account Management Module**: Users can successfully register, log in, and access their project records, ensuring secure data access.
      4. **ChatGPT Interaction Module User Experience Improvement**: ChatGPT suggestions are conveniently displayed for users, and the interface improvements align with user feedback, making it easy for users to apply these suggestions.
      5. **Data Visualization Improvement**: The UI enhancements provide clearer and more intuitive data displays, improving the overall user experience.
      6. **Project Demonstration and Delivery Preparation**: The demonstration video thoroughly showcases system functionality, and the GitHub project received review feedback from the external team and passed audit requirements.
4. **Incomplete Work**
   1. **Unfinished Items**:

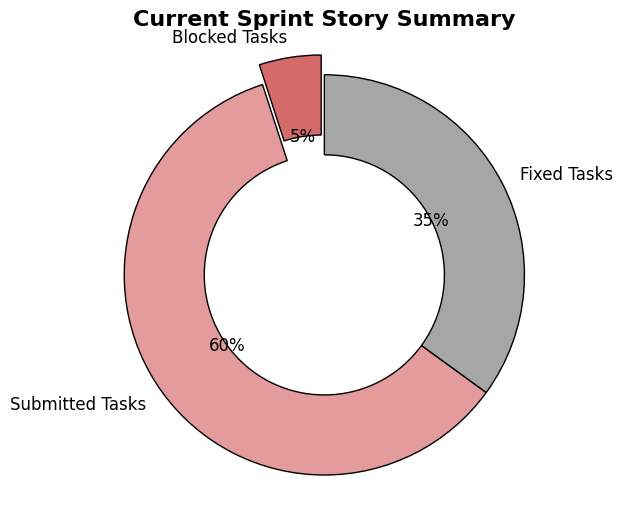
**Automated Testing and Debugging**: The development of automated testing tools has not yet been completed. Although integration testing has been conducted, the automated testing for some modules still needs to be refined. The primary reason for this delay is the complexity of dependencies encountered by the development team during multi-module integration, which caused delays in the development of test scripts.

* 1. **Deferred Items**:

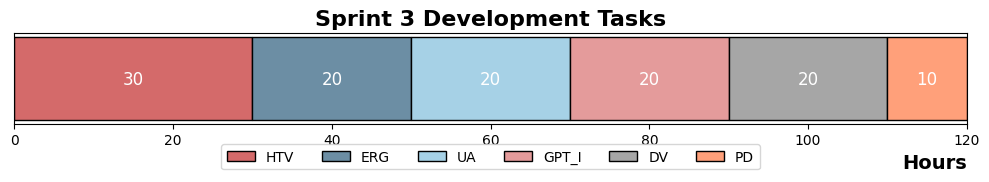
**Automated Testing and Debugging**:

**Integration Testing**: Some integration and automated testing tools have been postponed to the next stage to ensure test coverage and system stability.

1. **Challenges and Solutions**
   1. **Improving Context Clarity in ChatGPT Suggestions**
      1. **Challenge**: In optimizing the ChatGPT interaction module’s interface design, user feedback indicated that some suggestions lacked clear context, requiring users to spend time understanding and applying modifications. Users especially struggled to track each suggestion for code with multiple modifications.
      2. **Solution**: The team improved interface displays by adding detailed comments and providing step-by-step guidance, making it more intuitive for users to read and apply suggestions. Multiple rounds of user experience testing were conducted, with interface adjustments based on user feedback.
   2. **Cross-Module Coordination and Communication**
      1. **Challenge**: As several modules required seamless communication in the final integration phase, the team faced some communication challenges when handling cross-module dependencies and interface coordination, leading to inconsistencies during integration that impacted testing progress.
      2. **Solution**: The team addressed this issue by increasing regular internal communication meetings and implementing a cross-review process. After completing each module, developers from other modules participated in the review, ensuring a consistent understanding and implementation of interfaces and dependencies, reducing integration risks.
   3. **User Experience and Error Feedback Optimization**
      1. **Challenge:** During front-end and back-end interactions, the user interface often displayed incomplete content or animations when data loading was not yet finished, negatively affecting the user experience. Additionally, when data transmission failed, the system lacked a clear feedback mechanism, making it difficult for users and developers to troubleshoot issues.
      2. **Solution:** The front-end added loading state management indicators for various components, ensuring that the interface and animations were updated only after all related data had been fully loaded, preventing users from seeing incomplete interfaces during loading. Furthermore, for failed requests, the front-end implemented error-handling logic to provide users with clear error messages. On the back-end, detailed error logs were recorded to facilitate troubleshooting for developers. These optimizations not only enhanced the user experience but also made issue resolution more efficient.
2. **Review Feedback**
   1. **Summary of Sprint Review Feedback**:
      1. **Insufficient Automated Test Coverage**: In Sprint 3, although the team completed most module integration tests, automated test coverage remains insufficient, especially in scenarios involving complex business logic and system load. The delay in developing certain automated testing tools affected overall system stability verification.
      2. **Challenges in Module Integration**: During the integration phase, particularly when integrating the scoring calculation and ChatGPT response processing modules, the team encountered inconsistencies due to the complexity of dependencies and communication. The integrated behavior did not match expectations, causing delays and increasing testing and debugging workloads.
      3. **Further User Experience Improvement Needed**: Despite multiple optimizations to the ChatGPT interaction module interface, some issues remained unresolved based on user feedback. User experience, especially when handling multiple modification suggestions, was not as smooth as desired. Some users reported that the guidance was unclear when dealing with multiple suggestions, making it difficult to understand and apply the modifications.
   2. **Areas for Improvement**:
      1. **Increase Automated Test Coverage**: In future development, the team needs to refine the testing plan to expand automated test coverage for complex logic and scenarios. It’s essential to complete the automated testing tools on time in the integration phase and utilize more automated tools to reduce the workload and potential errors associated with manual testing.
      2. **Enhance Advanced Planning for Module Integration**: Module integration should be planned and scheduled more proactively. Small-scale integration testing should occur immediately after each module is developed to identify and resolve compatibility issues early, reducing rework and debugging pressure during the final integration phase. It is also recommended to consider detailed interface dependencies and prepare the testing environment within the integration plan to ensure smoother integration.
      3. **Continuous User Experience Optimization**: It is essential to continue gathering user feedback and optimizing the ChatGPT interaction module and other UI modules. More detailed user guidance, such as clear step-by-step instructions and comparison views, could enhance the user experience, helping users understand each modification suggestion. Additionally, incorporating real user scenario simulations during optimization could ensure the design aligns with actual user needs and usage habits.
3. **Areas for Overall Project Improvement and Optimization**
   1. **Increase Test Coverage and Efficiency for Automated Testing**: Although integration testing was completed in Sprint 3, automated test coverage remains inadequate, especially for complex logic and load testing. Future efforts should focus on further developing and optimizing automated testing tools to ensure comprehensive validation of all modules and their integrations. Advanced automated testing frameworks could be considered to increase testing efficiency and reduce the workload associated with manual testing.
   2. **Enhance User Experience and Interface Guidance**: While the ChatGPT interaction module has undergone several optimizations, user experience in scenarios with stacked suggestions and complex modifications still has room for improvement. Future considerations should include adding step-by-step guidance and detailed prompts to help users better understand and apply modification suggestions. In data visualization, a more intuitive interface—such as animated charts and interactive views—could improve user comprehension and engagement.
   3. **Further Enrichment of Data Visualization and Analysis**: The current history tracking and visualization module provides basic CWE trend and score change charts, but chart types could be further enriched. Adding more detailed correlation analyses and predictive models would help users better understand potential issues and future risks in their projects. Additionally, incorporating user-specific behavior analysis could provide personalized improvement suggestions.
   4. **Optimize Module Integration and Communication**: Strengthening integration planning between modules is crucial in future development. After completing each module, small-scale integration testing should be conducted immediately to minimize dependency issues during final system integration. Continuous Integration (CI) tools could be used to automate module integration, quickly identifying compatibility issues and enhancing overall system stability.
   5. **Enhance Security of User Authentication**: The current user authentication module supports basic registration and login functions, but multi-factor authentication (MFA) could be added to improve security. Options like SMS verification, email authentication, or mobile-based authentication apps should be considered to further safeguard user accounts.
   6. **Increase Support and Optimization for Large-Scale Projects**: For projects involving numerous external dependencies and complex code, the recent performance optimizations have improved processing efficiency, but further improvements could be made to the file analysis and database modules. Implementing distributed processing architectures or caching mechanisms could accelerate data retrieval and analysis. Additionally, the system should support more concurrent users to ensure stability under high-traffic conditions.
4. **Current Sprint Story Summary**



1. **Sprint 3 Development Tasks**



HTV (History Tracking & Visualization Module)

ERG (Export & Report Generation Module)

UA (User Authentication & Account Management Module)

GPT\_I (ChatGPT Interaction Module Improvement)

DV (Data Visualization Improvement)

PD (Project Demo & Delivery Preparation)

1. **Sprint 3 Burndown Chart with Ideal Projection**

